Center for Applied Autonomous Sensor Systems (AASS)
Örebro University, School of Science and Technology
SE-70182 Örebro
Sweden

OOP Lab-IV: Inheritance

More Cars and Vehicles

In this lab, we will look at several new concepts, using our previous code and projects.

Exercise 1: New Project

Create a new personal project on a Git-E server. Keep the project private and when, creating, tick the box to initialize the repository with README file, so you are able to clone it. Then, in NetBeans, clone your new repo to your development machine. Please, make a commit after every exercise.

Exercise 2: Vehicles

In a new project, copy over previously developed classes and add mode classes similarly to the UML diagram in Figure 1.

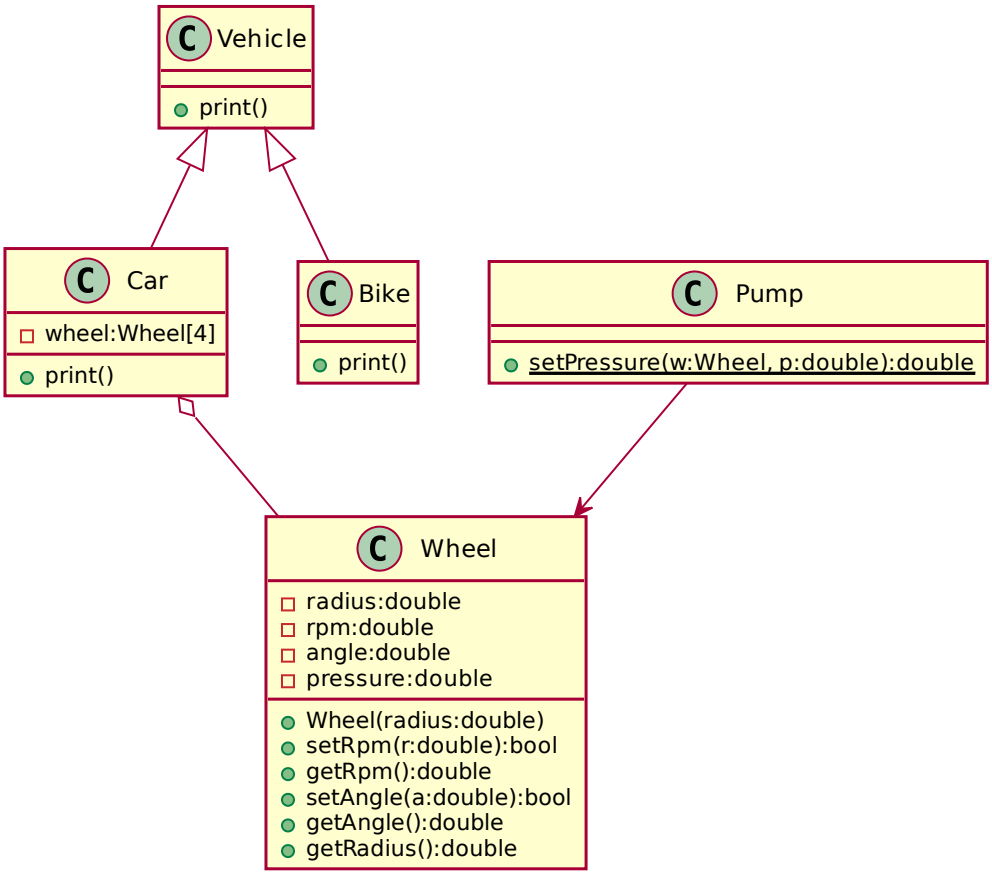

Figure 1: Vehices UML Diagram

In this diagram, Car and Bike inherit from Vehicle. All of them should contain definition of print function which is supposed to stream the object attributes out to the console stream (cout). In this function definitions, make each of them to output the name of a class (Vehicle, Car, or Bike).

Exercise 3: Testing

In your main.cpp file, create several vehicles. In C++, you can create objects of derived class with the type of the base class. For example:

```
Vehicle * v1 = new Car();
Vehicle * v2 = new Bike();
```

Create several objects in this way. Further in you code, use the print function to stream your objects class names into cout. If you didn't use any specific keywords, you should observe the same output for all functions. That is because of static function binding (use lecture notes and book chapter about static and dynamic bindings).

If you observe the same output (saying “Vehicle” in all cases) then you need to instruct your compiler to use dynamic binding. Please, look at your lecture notes, or a course book to find out how.

Exercize 4: Deadly Diamonds

In your program, create a new class "Boat" inheriting from Vehicle. After that, create a new class called "Amphicar", which inherits from both, Boat and Car classes (amphical is a floating car, so it has features of both). This will create a so called "diamond" inheritance (read more about it in your lecture notes). Resolve ambiguity in your code and make it working. Use your lecture notes and the course book to find out a way to do it.

Exercise 5: Streams and Files

In your project, create a new empty file and paste the following lines into it:

```
bike
amphicar
car
car
bike
car
amphicar
```

In the main program, open this file for reading. You will need to use the following libs:

```
#include <fstream>
#include <string>
```

Then, the file can be opened using:

```
ifstream inFile("vehicles.txt");
```

In the main function, make a while loop (while file is not ended) to **read every line from this file and instantiate objects**, depending on what word you read (bike, car, or amphicar). You can use a good() or eof() functions of the file stream.

To instantiate objects on run time, you need to pre-allocate a storage for your objects. The simplest solution is to create an array of pointers to the parent class and make its size big enough to accomodate vehicles from the file.

When you read from file, there is a useful function, which can read the whole string. It is called getline(). The are two signatures of this function:

```
istream& getline(istream& is, string& str);
istream& getline(istream& is, string& str, char delim);
```

The function accepts at least two arguments: which filestream to read from, and where to store the line. So, you have to prepare a std::string variable to store results. The latter function also allows to specify the line delimiter, if needed (might be useful for dealing with unknown OS).